Automatic Generation of Game Level Solutions as Storyboards

David Pizzi, Jean-Luc Lugrin, Alex Whittaker, and Marc Cavazza

***Abstract*—Game programmers rely on artificial intelligence techniques to encode characters' behaviors initially specified by game designers. Although significant efforts have been made to assist their collaboration, the formalization of behaviors remains a time-consuming process during the early stages of game development. We propose an authoring tool allowing game designers to formalize, visualize, modify, and validate game level solutions in the form of automatically generated storyboards. This system uses planning techniques to produce a level solution consistent with gameplay constraints. The main planning agent corresponds to the player character, and the system uses the game actions as planning operators and level objectives as goals to plan the level solutions. Generated solutions are presented as 2-D storyboards similar to comic strips. We present in this paper the first version of a fully implemented prototype as well as examples of generated storyboards, adapted from the original design documents of the blockbuster game *Hitman*.**

***Index Terms*—Artificial intelligence, games, multimedia computing, planning.**

I. Introduction

DESPITE the growing interest in interactive storytelling (IS) techniques, their actual applications to traditional gameplay design remain to be investigated. IS techniques are attracting much interest for their potential to develop new game genres but also as another form of procedural content generation, specifically dedicated to game events rather than objects or characters. However, many game designers have expressed concerns about the incorporation of such generative techniques in traditional game titles, mainly because of the lack of control they will have over dynamically generated content.

In the course of joint research with a major European publisher (Eidos Interactive, London, U.K.), a novel use for IS techniques was suggested, particularly as a support to the game design phase. It consists of generating all the possible solutions for a given game level considering the preservation of certain narrative content. This relies on the dual nature of IS planning technologies, which are able to generate sequences or narrative actions as well as various solutions to a game level problem.

Manuscript received March 19, 2010; revised July 22, 2010; accepted August 07, 2010. Date of publication August 26, 2010; date of current version September 15, 2010. This work was supported in part by the Department of Trade and Industry, via the Technology Programme BARDS Project, in collaboration with Eidos Interactive, Ltd. This manuscript is an extension to the paper previously presented at the Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE), Stanford, CA, October 2008.

D. Pizzi, J.-L. Lugrin, and M. Cavazza are with the School of Computing, Teesside University, Middlesbrough, TS1 3BA, U.K. (e-mail: d.pizzi@tees.ac.uk; j-l.lugrin@tees.ac.uk; m.o.cavazza@tees.ac.uk).

A. Whittaker is with WeRInteractive, London, EC1R 5DW, U.K. (e-mail: alexw@werinteractive.com).

Digital Object Identifier 10.1109/TCIAIG.2010.2070066

Traditionally, for multisolution-plan games such as *Grand Theft Auto*,[1] *Assassin's Creed*,[2] *Metal Gear Solid*,[3] or *Hitman*, the elaboration of alternative plan solutions relies on empirical methods, where solution variants mostly depend on the level designers' creativity. In this paper, in opposition to traditional whiteboard or flowchart, we propose a rational approach to the elaboration of level solutions based on knowledge representation and AI planning techniques capable of representing gameplay features. One of the major advantages of an AI-based approach is that it supports a systematic exploration of plausible solutions, and enables revalidation of a solution plan whenever the game environment is modified. In addition to supporting the generation of solutions from high-level principles, such an approach also constitutes a framework with which to evaluate plan complexity. Therefore, an AI-based authoring tool would help the designers to explore the possible completion plans as game level solutions, and to evaluate them in terms of difficulty and narrative experience.

However, solutions generated by AI planners are traditionally presented as lists of operators, which are notoriously difficult to read for the nonspecialist. By contrast, within most game companies (including Eidos Interactive), game designers use storyboards to represent and discuss alternative game level solutions. Therefore, to assist designers in generating, evaluating, and exploring plausible solutions in an intuitive way, we have adopted an interactive storyboard approach to allow an intuitive presentation and modification of the different level solutions produced by our IS planner. In the following sections, we illustrate system behavior and the authoring process with specific examples for a level of the released game title *Hitman Blood Money*.[4]

II. Related Work

Formal approaches have been previously proposed to model the design process of computer games [3], [6], [7], [20], [21]. The underlying idea is to model the spatio–temporal relationships which occur within the game universe. These techniques thus allow the description of the logical structure of the level missions in the game by modeling the ordering of action sequences using graphical models such as Petri Nets. The use of Petri Nets supports a dynamic visualization of game scenarios However, they are essentially designed for analyzing and validating preexisting scenarios rather than assisting in their creation.

[1]Grand Theft Auto by Rockstar Games: http://www.rockstargames.com/.

[2]Assassin's Creed by Ubisoft: http://www.assassincreed.com/.

[3]Metal Gear Solid by Konami: http://www.konami.com/.

[4]Hitman: Blood Money © 2006 IO Interactive A/S. Developed by IO Interactive A/S. Published by Eidos Interactive Ltd. All rights reserved.

In the field of IS itself, several authors have described tools facilitating the construction of interactive narratives using some visualization support. For instance, story graphs have been used in different authoring systems to explicitly represent all the possible story paths in INSCAPE [9], [29], U-Create [26], Scribe [19], and SceneMaker [13]. These tools present intuitive methods of visualization which can assist authors during the story creation process. However, they are based on nongenerative formalisms, constraining authors to manually encode all possible plan variations [22], [28].

From a different perspective, comics constitute a highly expressive medium, representing a story using a limited number of panels [18]. However, even though comic strips have been described as part of an IS system output in previous work [2], the creation of comics is quite a complex process and obeys a large set of rules (e.g., on transitions, panel shapes, etc.) [1], [10]. On the other hand, game level designers commonly use storyboards, which rely on simplified conventions. This is why, in order to visualize the solutions generated by our IS planner, we have replaced the standard planner's output by a more intuitive visual output using interactive comic-like storyboards. A planning approach also shared by Jhala *et al.* [15] with their intelligent storyboarding system generating 3-D movies from underspecified handcrafted 2-D storyboards.

III. Overview

Our system enables game designers to generate and explore complex game scenarios without relevant expertise in planning technologies, and immediately visualize and modify them through interactive storyboards. This approach differs from previous generative systems such as ScriptEase [8], which assists designers in producing character behavior scripts within the boundaries of previously validated game scenarios, in that our system produces level solutions from first principles. The basic philosophy is to generate various possible solutions to a given game level using the player character as the main agent, and gameplay actions as the basic elements of solution generation. This system uses heuristic search planning (HSP) [5] to generate level solutions, each legal game action being described as a planning operator. The description of the initial state, the level's objective as well as the level layout, constitute the input data. Other gameplay-related parameters for the simulation include the *Hitman*'s style, which influences the choice of certain actions and favors a certain style of solution (e.g., `stealth` versus `violent`). This is why, in addition to basic dimensions such as the actor's state, location, or event timeline, we also define a specific assassin style representative of this type of solution. The "killing" actions are then categorized according to the above style and the most discreet (i.e., the one with the predicate `style (stealth)` in their preconditions) will tend to require numerous subplans. The assassin style is set as an initial value and remains the same during the solution generation. As illustrated by Fig. 1, the exploratory approach supported by our system follows four main stages: *domain implementation* [Fig. 1(0) and 1(1)], *solution and storyboard generation* [Fig. 1(2)], *solution analysis* [Fig. 1(3)], and *solution modifications* [Fig. 1(4)].

The first stage corresponds to the elicitation of all knowledge required to describe a game level, such as the various states (e.g., the level goal is to kill three different targets, different initial states, etc.) and the various game actions described through their preconditions and their consequences. This step is carried out by the AI programmer(s) from the baseline scenario provided by the game designer(s). In terms of planning, this corresponds to *domain implementation*, which is constituted by the tuple $\langle \mathsf{O}, \mathsf{S}, \mathsf{G} \rangle$ where O is the set of available operators (i.e., the game actions), S is the current world state, and G is the goal (i.e., level objectives). The world state is represented by a conjunct of predicates defining the character's current beliefs, states, and goals as well as environment layout and events. The panel templates, which are used to generate storyboards, are attached to planning domain elements; therefore, they also need to be defined at this stage. Their composition will be detailed in the section *storyboard generation*.

The second step, *solution generation*, consists of generating a possible solution to the *Hitman* level under consideration. The world state is represented by a conjunction of predicates. We use HSP to generate level solutions from the initial state to the level goal [23]. It generates solutions to the game level as a sequence of actions leading to the level objective, represented as the plan's goal. We have opted for a "real-time" version of HSP by implementing RTA* [16] as its underlying search algorithm, which also allows anytime plan interruptions to test environment modifications. The heuristics are used to guide action selection towards the level solution. Their calculation is based on the simple value iteration (VI) method [17] and accounts for a significant fraction of the total central processing unit (CPU) time as classically described in the HSP literature [4]. The planner can produce on average a complete solution in approximately 1 s on a 2-GHz Intel processor, which is fully compatible with its use within a design/authoring environment. The *storyboard generation* is running in parallel to the *solution generation* phase. It uses templates to construct a storyboard panel from data corresponding to the action forming part of the current solution being generated. In turn, the various panels will be assembled sequentially into a storyboard presenting the complete level solution generated. As an alternative to offline generation of a complete solution, an online mode allows step-by-step generation of a solution, which includes the visualization of all possible outcomes. Starting from the initial state, the user can expand the plan at each step using an automatically generated tree representation until the goal state is reached. As illustrated by Fig. 2, after each action selected by the user, the system automatically offers a list of possible subsequent actions. Once completed, the solutions produced during the offline or online mode are analyzed. During *solution analysis* [Fig. 1(3)], the plan complexity is expressed in terms of the number of subplans identified. In addition, designers are allowed to interact and experiment with the storyboards as they are being produced, using *dynamic environment simulation* interfaces [Fig. 1(3)], which are described in detail in Section VI.

In later sections, we illustrate in detail each of these stages using real data[5] from the Casino level of the game title *Hitman*

[5]All design documents have been provided by IO Interactive Ltd., Copenhagen, Denmark.

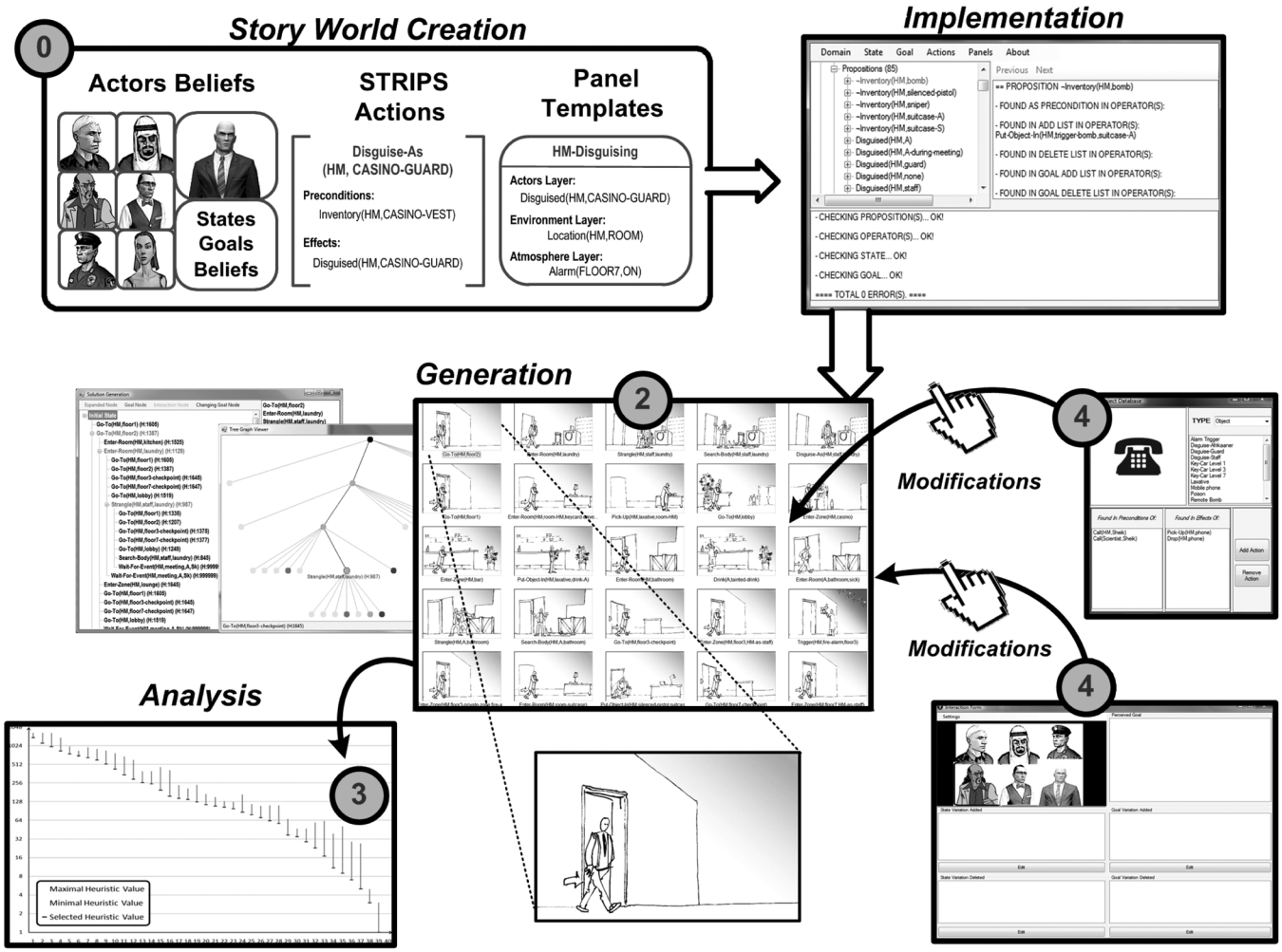

Fig. 1. Overview of the authoring process (see text for more details).

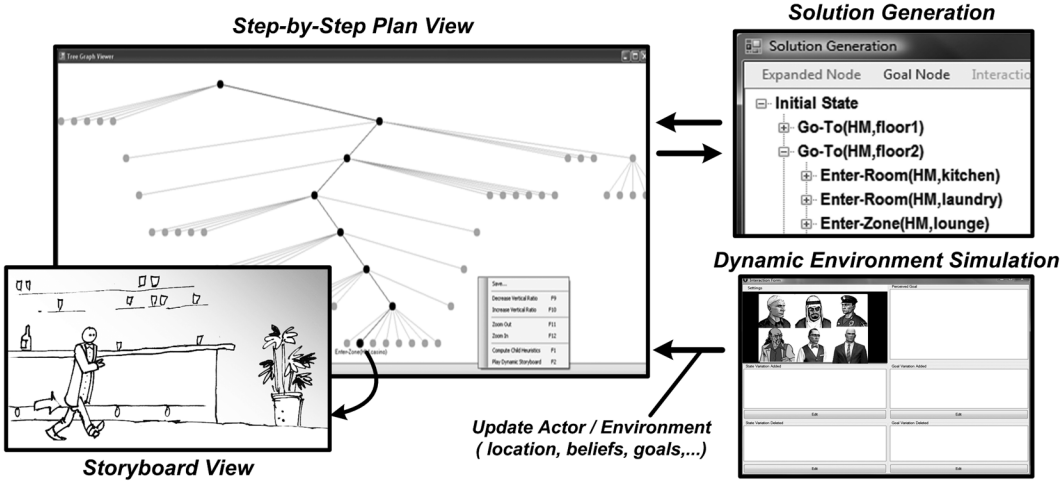

Fig. 2. Step-by-step solution generation proposing all possible alternative actions.

Blood Money. The story takes place in the Shamal Casino in Las Vegas, NV—a luxurious hotel/casino complex styled after the 1001 Nights theme. A Saudi sheik is about to meet with a South-African mercenary ("the Afrikaaner") at the luxurious casino to exchange some valuable materials. A scientist employed by the sheik will also be there to oversee the deal. The mission for *Hitman* is to assassinate all three men while remaining unnoticed. *Hitman* is a "stealth" game in the sense that the player is meant to incarnate a professional assassin where the more unnoticed the player remains, the greater score he will obtain. At the end of each level, the player's assassin style is expressed in terms of notoriety, which corresponds to discretion

and "accidental" appearances of target's deaths (for instance, leaving no witnesses will be qualified as a "stealth" assassin style). The different *Hitman* killing styles, reflecting his notoriety level, are separated into categories such as: `violent`, `classical`, `stealth`.

To avoid detection, gameplay includes specific actions such as disguising using enemy uniforms, hiding weapons or bodies in isolated rooms, or even attracting enemies' attention (throwing money, triggering alarms). In addition, players also have access to a large inventory of long-range weapon (e.g., sniper), as well as silent close range weapons (knives, poison). Expert players (i.e., `stealth` assassin) should be able to elaborate long and subtle plans, such as the one presented in Section IV, while beginners would envisage rather simpler or brute-force solutions. Therefore, one key aspect of the gameplay relies on the complexity and variety of possible solutions as a stealth approach may involve more careful planning, which also corresponds to a more sophisticated plan.

In this context, we first discuss the authoring methods for the domain implementation. We then describe the solution generation process while demonstrating the dynamic environment simulation through storyboard manipulation. Finally, we explain the dynamic creation of storyboards.

IV. Domain Implementation

The first step is to provide a complete propositional representation of the world (e.g., `disguised(HM, Afrikaaner)`, `location(HM, room)`, etc.). The initial state and the level goal are then simply represented by conjuncts of such propositions. The planning operators are represented using a STanford Research Institute Problem Solver (STRIPS)-like formalism (i.e., a set of propositions as preconditions and effects) [11] and correspond to game actions that can be performed by the player character *Hitman*. During generation, the selection of certain critical actions (e.g., various killing or neutralizing methods) is made using a categorization of operators according to the different *Hitman* styles. For instance, the `stealth` style will favor discreet actions such as `strangle` or `put-poison-in`, over noisier executions such as `shoot` or `trigger-bomb`.

The initial formalization of gameplay actions and states as a planning domain is not without impact on the types of solutions that will be generated. Common pitfalls are representing the domain at a level of abstraction too high or conversely using too specific actions, which will limit the generative power of the system [22]. Knowledge elicitation is required to identify and describe the actions for a level (e.g., `search-body`, `disguise-as`, `unlock-door`, or `shoot`). This is a manual process which can become error-prone when the size of the planning domain and the number of operators increase. It is well known that maintaining consistency between the predicates used by the various operators can become challenging when describing operators manually. The calculation of the HSP heuristic introduces further constraints, as it requires that each proposition appearing as a precondition of an operator should at least also be present in one add list (this could otherwise lead to the calculation of spurious heuristic values inducing potential action selection errors). Inconsistencies in predicates' labels could also be responsible for errors in the content of operators with other detrimental side effects. There is thus a need to check consistency of preconditions and effects every time changes to the planning domain are introduced as part of the knowledge elicitation process. Our authoring interface facilitates the development of a consistent planning domain by performing consistency checks and assisting the user in the exploration of the planning domain. The elicitation of a planning domain for our test level required one week, while the description of an additional level could be done in as little as two days by reusing common elements of the planning domain. This provides the basis for the generation of game level solutions to be validated by game designers, although in the first instance the solutions produced ignore problems arising from the fine-grained timing of certain actions. The latter point can be addressed through our storyboard image manipulation option, as described in Section VIII).

V. Solution Generation Process

The solution generation process operates under two modes: offline and online. In the former, the planner will consider the type of assassin selected by the level designer and output a complete storyboard proposing a solution whenever it exists. The latter, online, consists of a step-by-step plan simulation, in which the user can visualize the results using a tree-like hierarchy (Fig. 2). In this mode, level designers can construct the solution tree in a systematic fashion by manually selecting an action from among those proposed by the planner after each world state update. The designer can force the occurrence of an action despite a poor heuristic value and thus explore the results of alternative actions, which would not have normally been selected by the planner. This corresponds to an expert mode, which is also of interest when requiring detailed explanations for a generated solution. Solutions are generated according to the following steps. Every operator in the planning domain is tested for applicability. Whenever an operator's preconditions are satisfied, the operator is applied to create a new state that could further be extended. The shape of the entire tree can be examined, so that all possible solutions can be visualized. When a node is selected, the entire plan that leads to it can then be rendered as a storyboard using the *storyboard generation* process.

VI. Storyboard Generation

In *Hitman*, where solutions rely on a sophisticated plan elaboration, the causal relation between certain actions and the final goal may be difficult to perceive. This is mostly due to the lack of visibility for long-term causal dependencies. For instance, during the first part of a plan, *Hitman* may kill a random casino staff member to obtain his clothes or uniform (Fig. 5). This early sequence of actions will only make full sense when later on *Hitman* will exploit his staff appearance to serve a poisonous drink to one of his targets waiting at the bar. It thus appeared that the set of possible plans could be visually represented in order to easily understand and control the unfolding of the generated content.

Therefore, each game action, described by a planning operator, is associated with a panel template which supports the generation of the final image panel from a set of elementary images

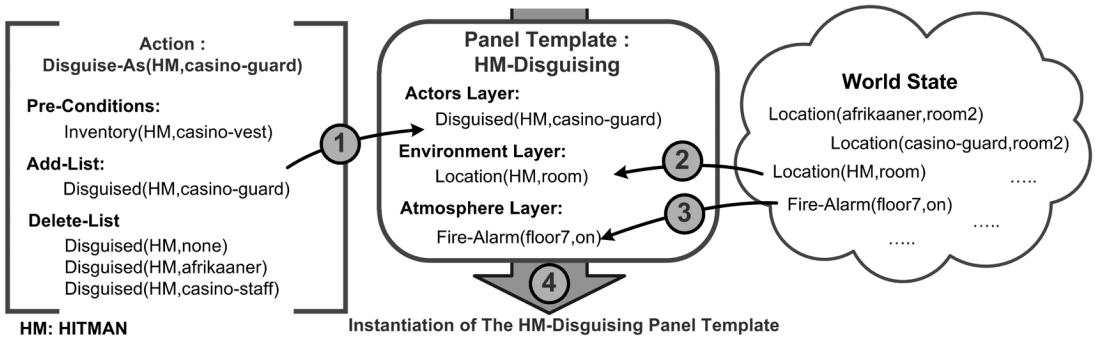

Fig. 3. Example of panel template instantiation from an action and current state.

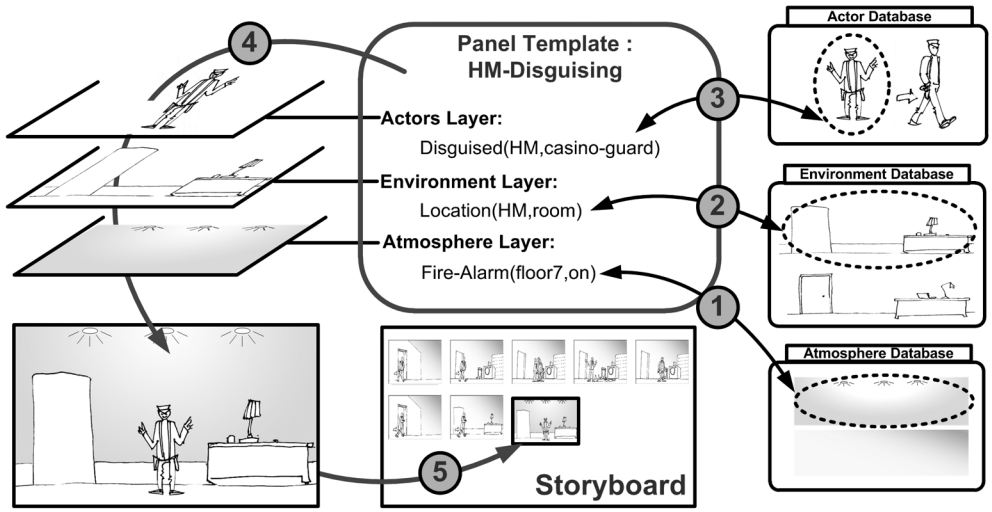

Fig. 4. Example of panel generation from an instantiated panel template.

(all original design documents have been provided by IO Interactive Ltd.). We use, as a way of dynamically creating graphical content, a method relying on image layer composition in which a storyboard image is composed of three main layers: 1) an atmosphere layer (background), 2) an environment layer (midground), and 3) an actors layer (foreground). For each layer, a template will define a position for an image (jpg, png, etc.). The atmosphere layer is used as a background image to emphasize a particular atmosphere following storyboard conventions (e.g., glows of fire or emergency lighting, etc.) or game events (e.g., fire alarm on), while the environment layer represents a given room or place such as a kitchen or a laundry. The actors layer is used to draw elements of the scene that could be rendered at different positions (e.g., characters, objects, etc.). The objects in the scene can be rendered at different preset positions (left, center, right, etc.) and different heights (above ground, ground level, etc.).

The panel generation process relies on two operations: panel template *instantiation* and *aggregation*. The first phase automatically takes place once an action has been selected as the next operation to perform. The generic template, previously associated with the action, is then activated; consequently, the actor layer is matched to the action add-list predicates [Fig. 3(1)] while the environment and atmosphere layers propositions are retrieved from the current world state [Fig. 3(2) and 3(3)]. Once instantiated with the current story context [Fig. 3(4)], the template is ready to be aggregated. During the panel aggregation phase, the template respectively browses the actor, environment, and atmosphere picture databases and selects the image associated to propositions previously identified [Fig. 4(1)]. For instance, if *Hitman* is actually located in a room, the proposition `location(HM, room)` is part of the world state, so the corresponding picture "Room.png" is selected [Fig. 4(2)]. One important gameplay feature is *Hitman*'s ability to disguise himself as other characters, which usually grants him access to particular level areas. Consequently, an additional selection process exists at the actor layer level to extract the correct picture corresponding to *Hitman*'s current appearance and movement (i.e., standing, walking, running, etc.).

The system first determines the current *Hitman* condition from the world state to then, extract the appropriate picture from the actor picture database, where a 2-D artist had previously associated a posture and a motion tag to each character picture. For instance, a casino guard actor has different pictorial representations in our databases (in one of these, he is standing while in the other he is walking [Fig. 4(3)]. Once each image layer has been selected, the final storyboard panel is rendered by drawing each layer from the furthest ones (i.e., the background and the location) to the closest ones (i.e., all the characters and objects) [Fig. 4(4)]. Finally, the rendered panel is added to the storyboard [Fig. 4(5)].

VII. Results and Validation

A. Overview

Based on the level design documents provided, we formalized entirely one of the actual levels of the game *Hitman Blood Money*, namely Mission 10 called "A house of cards." The resulting planning domain for this level contains 93 operators (i.e., planning actions) and 95 propositions. The length of the solution generated varies from 15 up to 50 actions, depending on the *Hitman*'s killing style (i.e., from `stealth` to `violent`). In terms of storyboard panel, the total number of panel templates is 24. However, we must take into account that certain templates cover a significant number of actions. For instance, the `HM-walking` template is used by 43 actions. The following sections briefly describe the storyboards generated to solve the level according to the assassin's desired style (i.e., `violent, normal, stealth`), and then demonstrate the exploration of alternative solutions through storyboard interactions and step-by-step plan generation.

B. The "Stealth" Assassin Solution

In this section, we illustrate one of the possible solutions generated under the `stealth` killing style, which usually generates the longest and the most elaborate plans. Figs. 5–8 illustrate critical parts of the generated 40-step plan in the form of its resulting storyboard. The level starts when *Hitman* arrives at the casino hotel entrance, where a room has been reserved to his name [Fig. 5(1): `check-in(HM, lobby)`].

Fig. 7 depicts how *Hitman* will execute and take over the Afrikaaner's identity [Fig. 7(14)], allowing him later to deceive the sheik's guards and infiltrate the meeting (Fig. 8). The first part of the plan is to isolate the Afrikaaner from the bar area by dressing as a waiter and serving him a tainted drink. In order to do that, the solution made use of an important gameplay feature, i.e., *Hitman*'s ability to change his appearance and disguise himself, which allows him to access specific restricted areas or to remain incognito.

In this solution, *Hitman* starts to search for an isolated staff member whom he could neutralize and steal his uniform, which in turn will grant him incognito access to most parts of the hotel. On the second floor, a waiter has been reported alone in the laundry room (note: this information has been given to the player during the mission briefing). After going up there [Fig. 5(2): `go-to(HM, floor2)` and Fig. 5(3): `enter-room(HM, laundry)`], *Hitman* strangles his victim silently, leaving no trace, an extra precaution specific to `stealth` style [Fig. 5(4): `strangle(HM, staff, laundry)`]. Then, he empties the waiter's pockets [Fig. 5(5): `search-body(HM, staff, laundry)`] and puts on his clothes [Fig. 5(6): `disguise-as(HM, staff)`]. Now, *Hitman*, disguised as a waiter, will serve the Afrikaaner a drink tainted with a powerful laxative [Fig. 6(11): `put-object(HM, laxative, drink-A)` and Fig. 6(12): `drink(A, tainted-drink)`] which would force him to urgently use the bar bathroom, where *Hitman* will discreetly murder him [`strangle(HM, A, storage)`], take his room keys [`search-body(HM, A, storage)`], and switch clothes with him [Fig. 7(14): `disguise-as(HM, A, storage)`].

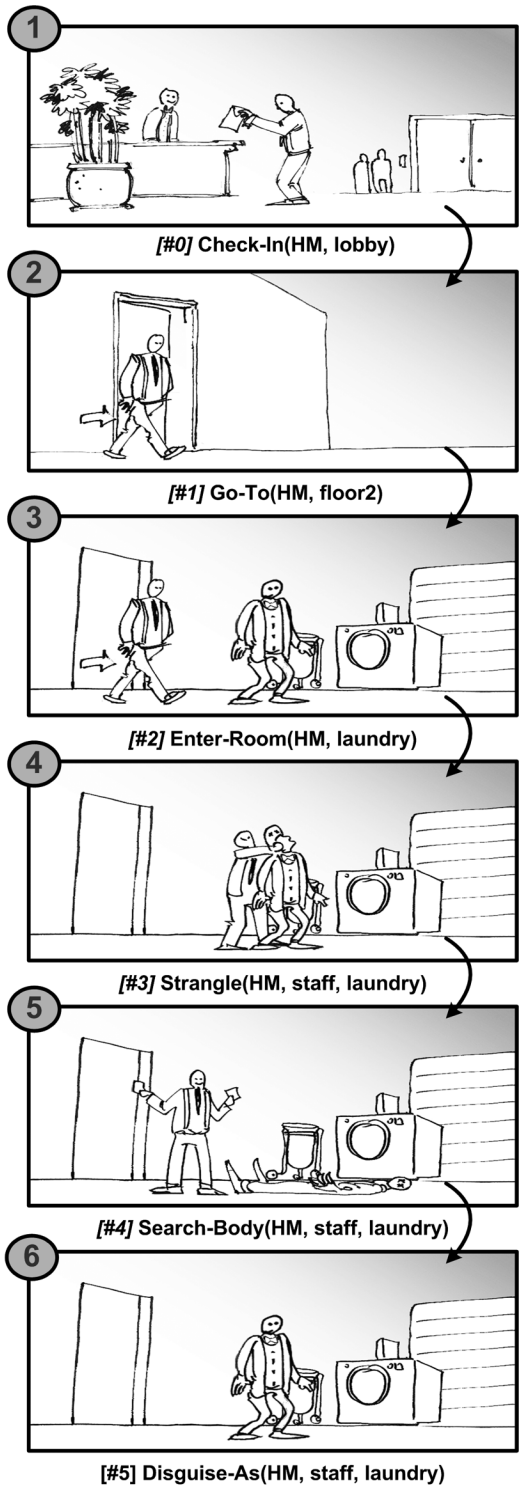

Fig. 5. Extract of a storyboard produced, which illustrates the subplan "acquire staff uniform." (See text for more details.)

In the meantime, the sheik arrives at the hotel entrance followed by his heavily armed bodyguards. In addition to the sheik's entourage, numerous security guards and cameras protect the Casino. In particular, the VIP Lounge, where the

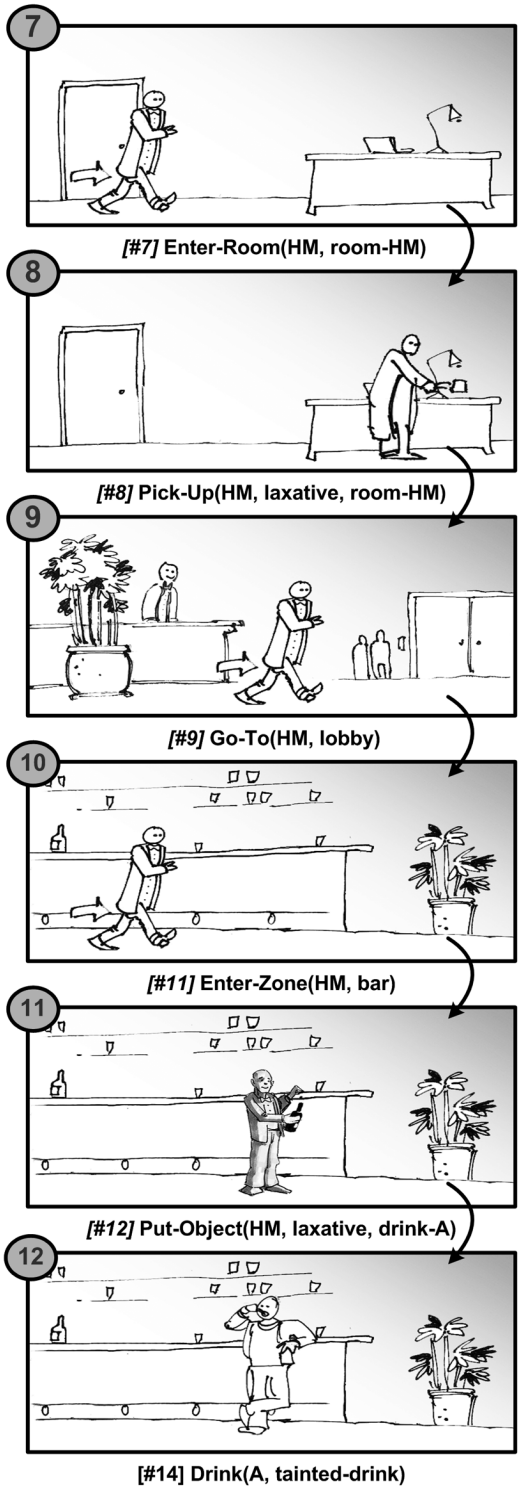

Fig. 6. Excerpt from a storyboard, which illustrates the subplan "attract Afrikaaner in bathroom." (See text for more details.)

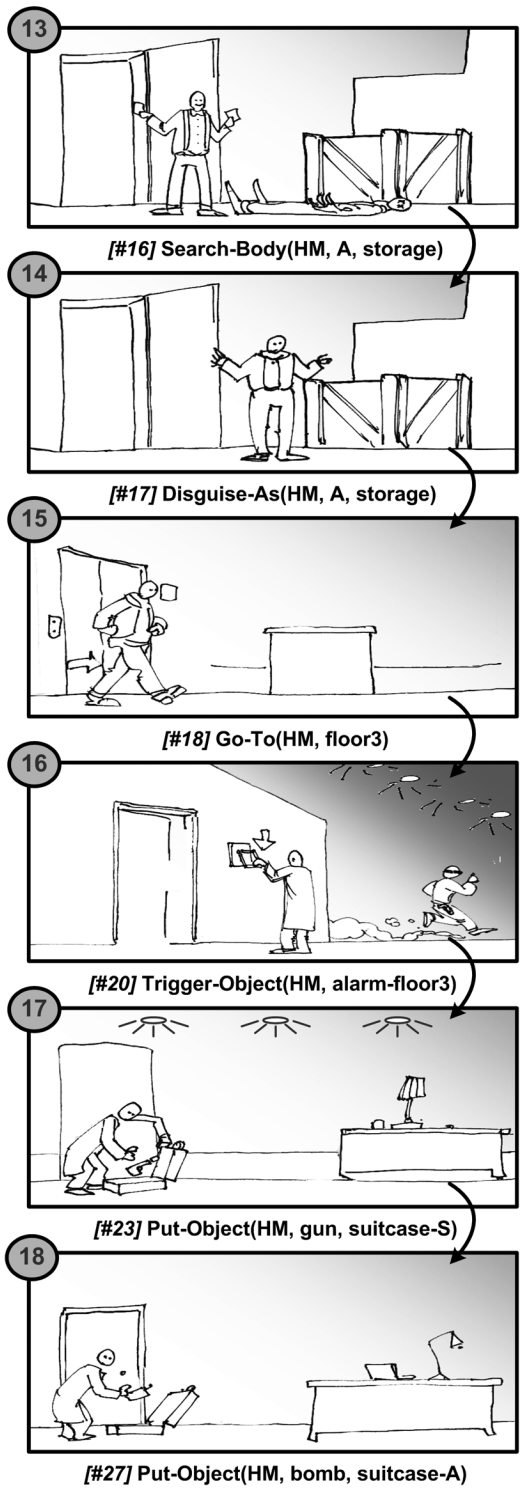

Fig. 7. Excerpt from a storyboard, which illustrates the subplan "booby-rrap sheik suitcase." (See text for more details.)

meeting is supposed to take place, is kept under severe surveillance; even waiters are searched before entering. In this context, bringing weapons within the meeting area without alerting the guards will require certain ingenuity.

One possible solution would be to place a weapon in one of the suitcases that would be exchanged during the meeting. As mentioned in the mission briefing, the Afrikaaner is supposed to bring a suitcase containing some secret genetic materials in order to exchange it for the diamond-filled suitcase brought by the sheik. In order to gain access to both the Afrikaaner's and the sheik's suitcases, first *Hitman* is going to trigger the hotel's fire alarm [Fig. 7(16): `trigger-object(HM, alarm-floor3)`] to force the immediate evacuation of everybody inside. He will then quickly drop a silenced gun

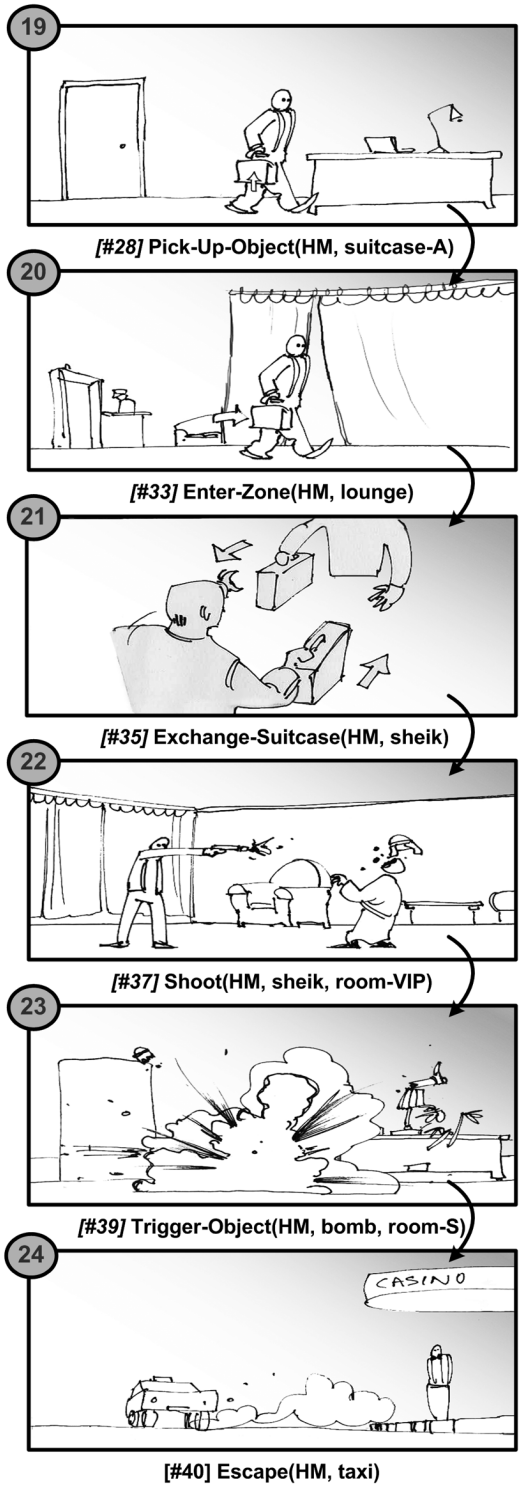

Fig. 8. Excerpt from a storyboard, which illustrates the subplan "kill-and-escape." (See text for more details.)

into the sheik's suitcase [Fig. 7(17): `put-object(HM, gun, suitcase-S)`], and a booby-trapped bomb within the Afrikaaner's suitcase [Fig. 7(18): `put-object(HM, bomb, suitcase-A)`].

Disguised as the Afrikaaner and carrying no weapon, *Hitman* now has no trouble passing the security checkpoint [Fig. 8(20): `enter-zone(HM, lounge)`] and accessing the private room via the VIP lounge. The sheik welcomes his guest and orders two of his men guarding the Scientist to bring him the diamond-filled suitcase. *Hitman* exchanges suitcases with the sheik [Fig. 8(21): `exchange-suitcase(HM, sheik)`]. As soon as the bodyguards leave with the Afrikaaner's suitcase, *Hitman* retrieves the silenced pistol from the suitcase the sheik has just given him [`pick-up-object(HM, gun, suitcase-S)`] and shoots him in the head [Fig. 8(22): `shoot(HM, sheik, room-VIP)`]. *Hitman* then walks out of the lounge and heads for the exit. Meanwhile, bodyguards bring the Afrikaaner's suitcase to the Scientist. The Scientist is blown to pieces as soon as he opens the booby-trapped suitcase [Fig. 8(23): `trigger-object(HM, bomb, room-S)`]. The mission objectives now being completed, *Hitman* can peacefully leave the hotel as his targets have been killed without raising any suspicion on his presence. *Hitman* can finally escape the hotel using a taxi parked in front of the entrance [Fig. 8(24): `escape(HM, taxi)`].

TABLE I
RESULTS

Assassins' style	Plan Length Range (Number of Operators)	Sub-Plan Count Range
"`Stealth`"	35-50	12-17
"`Classical`"	20-25	6-10
"`Violent`"	15-20	4-7

Table I provides an overview on the range of plan lengths and counts of subplans for each style. This notion of subplan is discussed in Section IX. The variation of the plan length corresponds to tests involving different initial world state (changing the *Hitman*'s default inventory by adding a sniper for instance) or reflecting modification experimentation (adding, removing objects). For each assassin style, our planner generated solution plans very similar to the ones originally planned by the level designer, or the ones provided in published strategy guides or online spoilers. Notwithstanding minor timing issues, all solutions computed have been successfully tested in the game. In a certain sense, our results validate the quality of the solutions computed by our planner, as well demonstrating the generative aspect of IS as a technique for game solution exploration.

However, the planner and its storyboards by themselves do not simulate the environment's dynamics (the continuous motions of nonplayer characters during the game), and therefore, do not permit the measurement of the impact of (sudden) environmental changes on a possible solution. Later sections discuss how to address this specific limitation.

VIII. Solution Modification and Revalidation

Here the planning agent only considers the actions of the player character (i.e., *Hitman*). However, in the dynamic game environment the nonplayer characters (NPCs), including *Hitman*'s targets, have their own game-related autonomous behaviors. For instance, the pseudorandom movement of the NPC inside the facilities could cause our `stealth` plan to fail due to the absence of the waiter in the laundry room [Fig. 5(3)]. Additionally, *Hitman*'s targets usually have sophisticated behaviors leading them to move constantly through the level

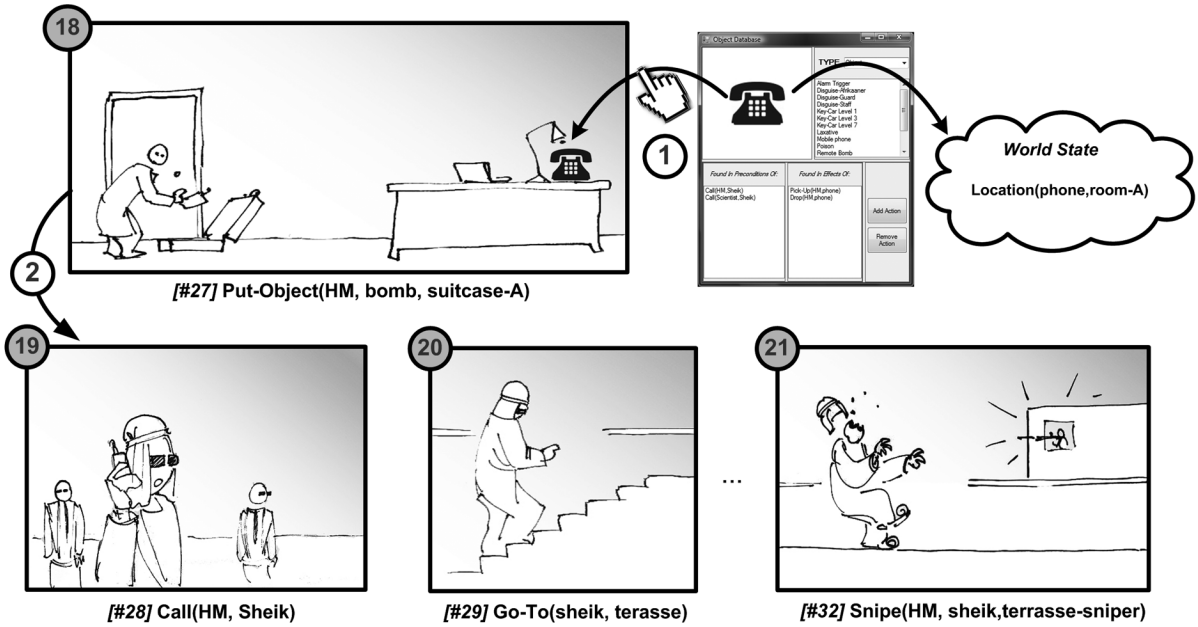

Fig. 9. Example of world modification by storyboard's panel alteration (drag-and-drop a phone item into the Afrikaner's room providing in term a novel plan to assassinate the sheik).

as they pursue their own goals. For instance, the Afrikaner keeps walking cyclically from the bar to his room passing by the bathroom providing several options to kill him and each of these options could be represented in the scenarios. In a highly dynamic world, where AI-controlled characters could generate events or provide/substitute items at any time, many plans could suddenly become obsolete. Therefore, in order to simulate world dynamics, spatio–temporal variations have to be simulated within the solution generation process. Subsequently, we have introduced the possibility for the designer to modify the world state at any stage in order to reproduce dynamic state variations that will normally occur within the game (e.g., simulating NPC movements from one room to another). Our plan generation is therefore synchronized with plan interruptions, so as to explore opportunities for various player and NPC actions.

Our dynamic environment simulation interfaces (i.e., agent and object databases) offer views of each current actor's beliefs, goals, and location as well the possibility to modify them at anytime while generating the plan in a step-by-step fashion (Fig. 2), or when visualizing the final storyboard (Fig. 1). Through these graphical interfaces, users can drag-and-drop item's pictures directly on the storyboard to modify the environment, and immediately visualize the impact of such modifications on a proposed solution plan.

A. World Modification

Within our system, the user can simulate changes in the environment or user timing failures by directly altering the storyboard's panel (a kind of "visual programming"), or by modifying the planner's world state. The first method offers an intuitive way for non-AI experts to modify the world state at any time during plan elaboration by simply inserting/removing objects from the storyboard's panel. Using our object database interface, users can drag-and-drop objects' pictures in the actor's layer of the panel (Fig. 9) and instantaneously visualize their impact on the solution previously proposed. For instance, a user could insert a telephone's item in one of the storyboard's image [Fig. 9(1)], to trigger the replanning of the entire plan [Fig. 9(2)]. To support such interactions, planner's actions (i.e., operators) are previously associated to actor's pictures, and update the world state based on the context of the storyboard panel. The insertion of the phone's picture over the Afrikaaner's room will immediately update the planner's world state by adding the following proposition: `location(phone,room-A)`. As a direct consequence of this modification, an entirely new plan is generated taking into account the new affordances provided by such an object (e.g., attracting the sheik outside to answer to the Afrikaner's call will give an occasion for *Hitman* to snipe him from a hotel room [Fig. 9(19)–(21)].

Meanwhile, users can also modify characters or objects that are not represented by the storyboard. For instance, users could alter any NPC agents' location, or trigger any world events at a certain time by simply clicking on a panel and using our agent database interface to modify a predicate. For instance, to reproduce characters movements, a user can update a proposition within the NPC actions such as `update-position (Afrikaaner, room2)` or world events such as `character-level-arrival (sheik, lobby)`.

B. Timing Issues

Another critical aspects of "Stealth"-type games are timing issues. In such games, accurate timing between actions is an issue that players often have to face. Consequently, level designers might want to explore unexpected consequences of timing failures when considering a particular solution. For example, the player may have been waiting too long and is now unable to interfere with the unfolding of the story. In the previously presented solution, we have seen *Hitman* placing a

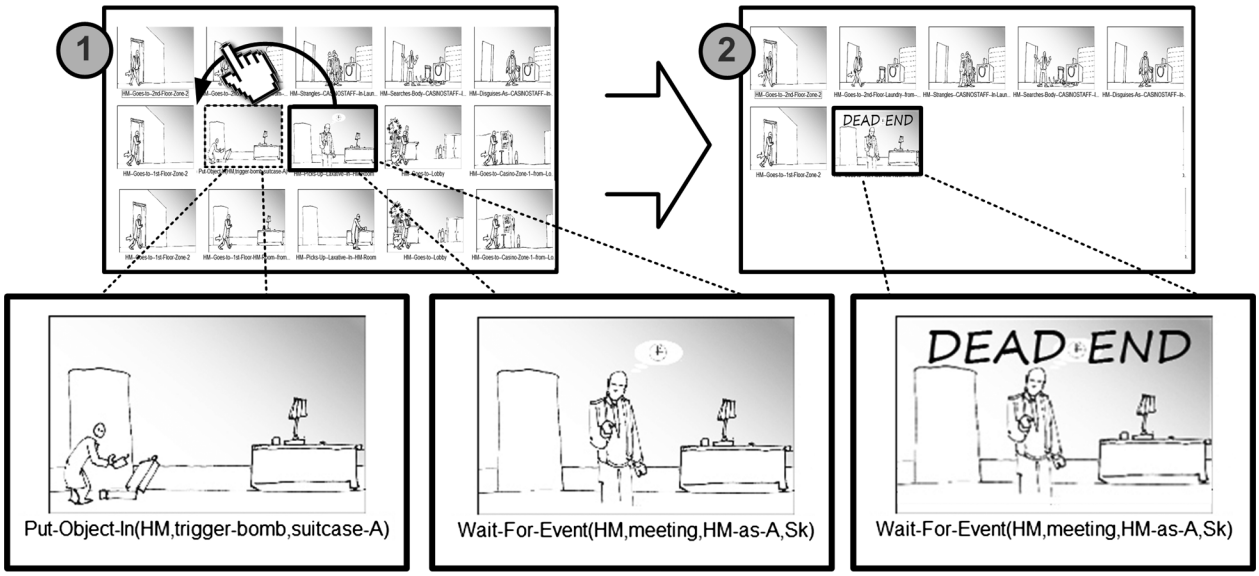

Fig. 10. Example of timing issue simulation by storyboard's panel reordering (here user drag-and-drop the panel "waiting-for-event" before the "put-object-in" one, which is artificially creating a delay leading to a plan failure).

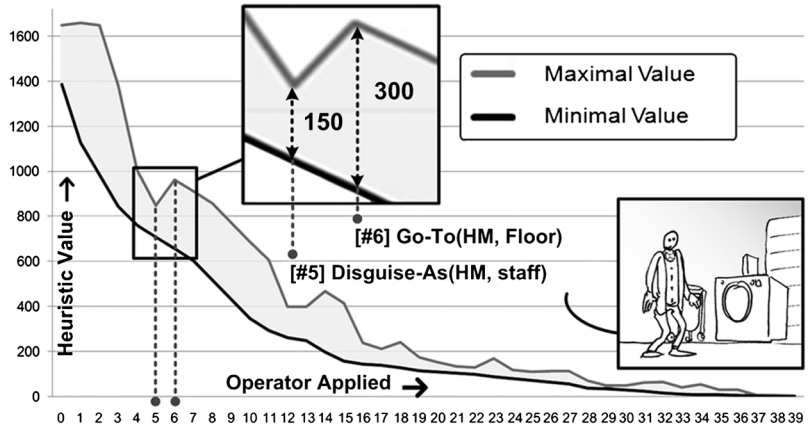

Fig. 11. Evolution of the HSP heuristic during plan execution.

bomb inside the Afrikaaner's suitcase [Fig. 7(18)]. Now, the designer might want to consider what will happen if the player arrives too late in the Afrikaner's room (i.e., while the meeting has already started) and thus cannot find the suitcase.

Fig. 10 depicts how the designer will simulate such situation by simply inserting (or moving) a `wait-for-event` action earlier in the storyboard sequence before the `put-object-in` action [Fig. 10(1)]. By forcing the meeting to start earlier, the designer prevents the planner selecting operators that can only occur before the meeting. This provokes a dead end, as the suitcase is now no longer accessible, forcing the solution to end prematurely. In this example, a solution has been interrupted by simulating a timing issue, which created a dead end. In such cases, the planner will signal a plan failure by displaying a "dead-end" message on the action triggering the failure [Fig. 10(2)]. In these short examples, we demonstrated how intuitive and expressive combinations of storyboard generation and modification could help to understand complex solutions, explore alternatives, and immediately perceive dead ends.

IX. Analysis

Due to the plans' complexity and size, the evaluation of their quality along with the exploration of alternative solutions can become a rather tedious task. To address this problem, our interface presents a tool assisting plan evaluation and exploration. This tool extracts and signals "key" actions from a generated plan, and so gives the possibility to level designers to explore the impact of a world state change before or after these critical actions. The identification of critical actions relies on a post-hoc analysis of the maximum and minimum heuristic evolution and more particularly the extraction of significant variations among the values computed. As illustrated by Fig. 11, each action chosen by the planner naturally contributes to an overall heuristic decrease during the plan progression. Consequently, no significant distinction can be directly extracted from the heuristic evolution. However, it can be seen that the range of values between the minimal and maximal heuristics varies considerably during story progression. In fact, even if the minimal

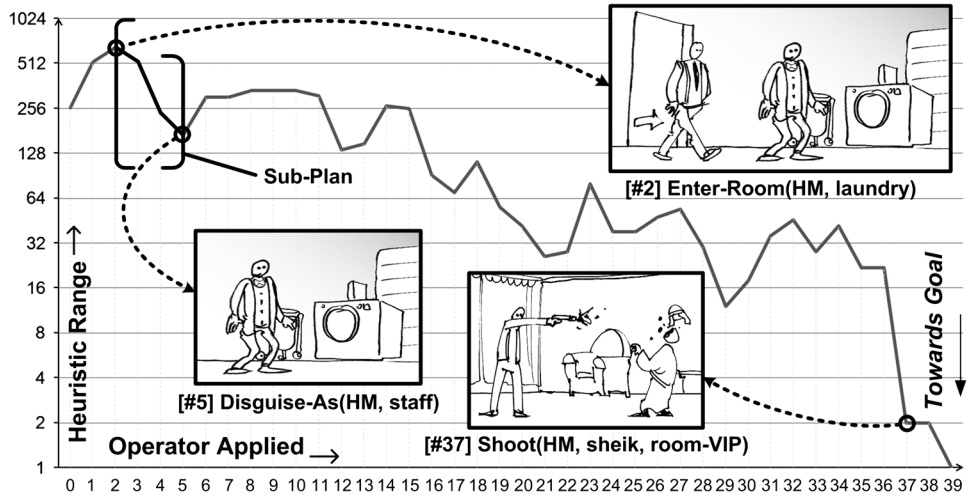

Fig. 12. Subplan identification from heuristic range variation (here: subplan "acquire staff uniform" from operator #2 to #5 has been identified).

values are naturally always decreasing, the maximal values fluctuation is noticeable after the application of certain actions.

As depicted on Fig. 11, the heuristic variation gap can nearly double in value between two subsequent actions (i.e., operator #5 and #6). The steep increase appearing right after operator #5 has been applied signifies that a larger population of possible actions has become available (i.e., perplexity). Here, the casino-staff uniform acquired by *Hitman* at action #5 opens a novel plan perspective by permitting a novel set of actions. As explained in the rest of the section, such a steep change could correspond to a landmark action denoting the end of a subgoal and the commencement of a novel one. In our example, the first key actions identified correspond to actions #2 and #5. Between those two operations, the heuristic range denotes a major decrease, which could be interpreted as a major progress realized by *Hitman* toward his final goal. It could also be perceived as the achievement of a subgoal, which corresponds here to the acquisition of an appropriate set of clothes for the rest of the plan. This episode corresponds to the action sequence starting at action #2 and finishing after action #5 (see Fig. 12) where *Hitman* enters the laundry, kills the staff member, and finally disguises as him. The action #37 is another example of a significant achievement highlighted by a sudden range variation once applied, when *Hitman* finally shoots his last target (Fig. 12).

In the case of an extensive plan containing over 30 actions, this system brings the level designer's attention to intermediate or critical achievements. As a direct consequence, designers could then experiment with alternative plans by modifying the world state necessary for these key actions using the dynamics environment simulation or domain implementation authoring interfaces. In addition, the number of subgoals identified within a certain plan provides pertinent indicators of the level solution difficulty. It gives level designers opportunities to relax or reinforce plan conditions to improve the gameplay experience. Our notion of subgoal recognition appears to be related to landmarks identification in traditional planning [14], which reflect critical intermediary plan subtasks. Once identified, landmarks can be used to decompose the original problem by focusing the search toward the achievement of these subgoals rather than targeting the final goal. However, landmark extraction relies on complex backward-chaining techniques [24] and, despite certain improvements, can produce dead-end solutions (see [25]).

Here, we have proposed a method to identify critical plan actions, and identify subplans, by relying on heuristics range analysis, with primary intention to help level designer to rapidly estimate and balance the overall gameplay difficulty in term of subplan amount.

X. Conclusion

While most of the AI techniques used in computer games are still dedicated to character behavior generation, we have presented an AI system assisting the game design process. This approach, derived from IS technologies, supports the generation of level solutions, while validating their narrative content. Furthermore, our system also supports the exploration of alternative solutions or dead ends by direct interaction with the generated storyboards. Our authoring environment assists domain formalization, solution plan visualization, modification, and evaluation, allowing game designers to preview solution complexity and adjust game balance. Our first prototype has been developed in collaboration with Eidos Interactive and tested using data from the released title *Hitman Blood Money*. Overall, our approach is applicable to all those games which offer a large variation of solutions depending on a different use of environment's resources, within the same gameplay constraints. This is mostly a characteristic of adventure games with rich environments and multiple opportunities for interaction and use of game objects, which support different playing "styles." Most recent releases of adventure games tend to fall into this category. Although no systematic or quantitative evaluation of the solution generation system has been undertaken, we have carried out a qualitative analysis of the solution produced, both internally and by submitting level solutions to the original game design team. Our qualitative analysis consisted in the validation of individual solutions by actually playing the game according to their course of actions, as well as identifying similar solutions in existing strategy guides or online game spoilers. When reporting the complete set of solutions to designers at Eidos, it

appeared that the system had produced a few instances of original solutions not hitherto identified by the design (or testing) teams, which constituted the best confirmation of its original objectives. However, our future work will aim at facilitating the identification of "fragile" plans (i.e., easily derailed due to their reliance on a delicate timing or use of resources (e.g., missing a critical event such as the sheik and Afrikaner meeting, or simply running out of ammunition for specific weapons). We are thus currently investigating techniques to simulate spatio–temporal variations of an action sequence, and their corresponding impact on plan execution. One research direction consists in altering world states in a systematic way around landmark actions (by automatically adding or removing object or character from certain locations and measuring their impacts). Using AI in support to game design appears to be a promising research topic [12], [27], which can have an impact on development costs and could facilitate collaboration between AI programmers and game designers in the workplace.

ACKNOWLEDGMENT

The authors would like to thank IO Interactive Ltd. for providing the original storyboards and design documents. They would also like to thank J. Merceron of Eidos Interactive (now Square Enix) for his support during this research project.

REFERENCES

[1] T. Alves, A. McMichael, A. Simões, M. Vala, A. Paiva, and R. Aylett, "Comics2D: Describing and creating comics from story-based applications with autonomous characters," in *Proc. Int. Conf. Comput. Animat. Social Agents*, Hasselt, Belgium, 2007, pp. 79–86.

[2] T. Alves, A. R. Simões, R. Figueiredo, M. Vala, A. Paiva, and R. Aylett, "So tell me what happened: Turning agent-based interactive drama into comics," in *Proc. 7th Int. Joint Conf. Autonom. Agents Multiagent Syst.*, Estoril, Portugal, 2008, pp. 1269–1272.

[3] D. Balas, C. Brom, A. Abonyi, and J. Gemrot, "Hierarchical Petri Nets for story plots featuring virtual humans," in *Proc. 4th Artif. Intell. Interactive Digital Entertain.*, Stanford, CA, 2008, pp. 2–9.

[4] B. Bonet and H. Geffner, "Planning as heuristic search: New results," in *Proc. Eur. Conf. Planning*, 1999, pp. 360–372.

[5] B. Bonet and H. Geffner, "HSP: Heuristic search planner," *AI Mag.*, vol. 21, no. 2, pp. 13–33, 2000.

[6] C. Brom and A. Abonyi, "Petri-Nets for game plot," in *Proc. Artif. Intell. Simul. Behav.*, Bristol, U.K., 2006, vol. 3, pp. 6–13.

[7] F. Collé, R. Champagnat, and A. Prigent, "Scenario analysis based on linear logic," in *Proc. ACM SIGCHI Int. Conf. Adv. Comput. Entertain. Technol.*, Valencia, Spain, 2005, article no. 1.

[8] M. Cutumisu, D. Szafron, J. Schaeffer, K. Waugh, C. Onuczko, J. Siegel, and A. Schumacher, "A demonstration of ScriptEase ambient and PC-interactive behavior generation for computer role-playing games," in *Proc. 2nd Artif. Intell. Interactive Digital Entertain. Int. Conf.*, Marina Del Rey, CA, 2006, pp. 141–142.

[9] M. Dade-Robertson, "Visual scenario representation in the context of a tool for interactive storytelling," in *Lecture Notes in Computer Science*, M. Cavazza and S. Donikian, Eds. Berlin, Germany: Springer-Verlag, 2007, vol. 4871, pp. 3–12.

[10] W. Eisner, *Comics & Sequential Art*. Tamarac, FL: Poorhouse, 1985.

[11] R. Fikes and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," in *Proc. 2nd Int. Joint Conf. Artif. Intell.*, 1971, pp. 608–620.

[12] D. Fu and R. Houlette, "Putting AI in entertainment: An AI authoring tool for simulation and games," *IEEE Intell. Syst.*, vol. 17, no. 4, pp. 81–84, 2002.

[13] P. Gebhard, M. Kipp, M. Klesen, and T. Rist, "Authoring scenes for adaptive, interactive performances," in *Proc. 2nd Int. Joint Conf. Autonom. Agents Multiagent Syst.*, Melbourne, Vic., Australia, 2003, pp. 725–732.

[14] J. Hoffmann, J. Porteous, and L. Sebastia, "Ordered landmarks in planning," *J. Artif. Intell. Res.*, vol. 22, pp. 215–278, 2004.

[15] A. Jahala, C. Rawls, and M. R. Young, "Longboard: A sketch based intelligent storyboarding tool for creating machinima," in *Proc. Florida Artif. Intell. Res. Soc. Conf.*, 2008, pp. 386–391.

[16] R. E. Korf, "Real-time heuristic search," *Artif. Intell.*, vol. 42, no. 2–3, pp. 189–211, 1990.

[17] Y. Liu, S. Koenig, and D. Furcy, "Speeding up the calculation of heuristics for heuristic search-based planning," in *Proc. 18th Nat. Conf. Artif. Intell.*, 2002, pp. 484–491.

[18] S. McCloud, *Understanding Comics*. Northampton, MA: Kitchen Sink, 1993.

[19] B. Medler and B. Magerko, "Scribe: A tool for authoring event driven interactive drama," in *Lecture Notes in Computer Science*, S. Göbel, R. Malkewitz, and I. Iurgel, Eds. Berlin, Germany: Springer-Verlag, 2006, vol. 4326, pp. 139–150.

[20] S. Natkin and L. Vega, "Petri net modelling for the analysis of the ordering of actions in computer games," in *Proc. 4th Int. Conf. Intell. Games Simul.*, London, U.K., 2003, pp. 82–92.

[21] S. Natkin, L. Vega, and S. Grünvogel, "A new methodology for spatiotemporal game design," in *Proc. 5th Game-On Int. Conf. Comput. Games, Artif. Intell. Design Edu.*, Q. Mehdi and N. Gough, Eds., Reading, U.K., 2004, pp. 109–113.

[22] D. Pizzi and M. Cavazza, "From debugging to authoring: Adapting productivity tools to narrative content description," in *Lecture Notes in Computer Science*, U. Spierling and N. Szilas, Eds. Berlin, Germany: Springer-Verlag, 2008, vol. 5334, pp. 285–296.

[23] D. Pizzi, F. Charles, J.-L. Lugrin, and M. Cavazza, "Interactive storytelling with literary feelings," in *Lecture Notes in Computer Science*, A. C. R. Paiva, R. Prada, and R. W. Picard, Eds. Berlin, Germany: Springer-Verlag, 2007, vol. 4738, pp. 630–641.

[24] J. Porteous, L. Sebastia, and J. Hoffmann, "On the extraction, ordering, and usage of landmarks in planning," in *Proc. 6th Eur. Conf. Planning*, A. Cesta and D. Borrajo, Eds., 2001, pp. 37–48.

[25] S. Richter, M. Helmert, and M. Westphal, "Landmarks revisited," in *Proc. 23rd AAAI Conf. Artif. Intell.*, 2008, pp. 975–982.

[26] S. Sauer, K. Osswald, X. Wielemans, and M. Stifter, "U-Create: Creative authoring tools for edutainment applications," in *Proc. 3rd Technol. Interactive Digital Storytelling Entertain.*, Darmstadt, Germany, 2005, pp. 163–168.

[27] A. M. Smith, M. J. Nelson, and M. Mateas, "Ludocore: A logical game engine for modelling videogames," in *Proc. IEEE Conf. Comput. Intell. Games*, 2010, to be published.

[28] U. Spierling, "Adding aspects of implicit creation to the authoring process in interactive storytelling," in *Proc. 4th Int. Conf. Virtual Storytelling*, Saint-Malo, France, 2007, pp. 13–25.

[29] N. Zagalo, S. Göbel, A. Torres, and R. Malkewitz, "INSCAPE: Emotion expression and experience in an authoring environment," in *Proc. 3rd Technol. Interactive Digital Storytelling Entertain.*, Darmstadt, Germany, 2006, pp. 219–230.

David Pizzi received the B.Sc. degree (first class) in computer games programming from Teesside University, Middlesbrough, U.K., in 2006.

He is a Research Fellow in the Intelligent Virtual Environment Lab, Teesside University, Middlesbrough, U.K. He is currently working in the field of interactive storytelling, emotional planning for character-based interactive storytelling, planning technologies, and authoring methods.

Jean-Luc Lugrin received the M.Sc. degree in computer graphics from Teesside University, Middlesbrough, U.K., in 2002.

Currently, he is a Principal Lecturer in Games Programming at Teesside University. His research focuses on knowledge representation and new behavioral approaches for virtual reality as well as immersive displays. He works on AI-based world behavior for emergent narratives.

Alex Whittaker received the B.Sc. degree in genetics from University College London, London, U.K., in 1989 and the A.M.Sc. degree in artificial intelligence from Queen Mary, University of London, London, U.K., in 1998.

He then moved into bioinformatics working with the Cancer Research U.K. and then Glaxo SmithKline, and then moved into the games industry working first with Sony Psygnosis on Playstation titles. He went on to work with several other developers including a significant time with Eidos on the Championship Manager franchise. He has maintained a strong interest in interactive storytelling and is now working in that field for WeRInteractive, London, U.K.

Marc Cavazza received the Ph.D. degree in biomathematics from the University of Paris 7, Paris, France, in 1991.

Currently, he is a Professor at Teesside University, Middlesbrough, U.K., where he has been working on interactive storytelling with his research group since 2000. He has authored more than 200 papers in the field of virtual agents and intelligent user interfaces.

Mr. Cavazza was a Program Committee Co-Chair for the ACM Intelligent User Interfaces and the ACM Multimedia (Human-Centered Track) in 2010.